SAGE-QA: Neurosymbolic Support-Aware Graph Evidence Retrieval for Question Answering

Julie Loesch[1], Michel Dumontier[1] and Remzi Celebi[1]

[1]Department of Advanced Computing Sciences, Maastricht University, Paul-Henri Spaaklaan 1, Maastricht, 6229 EN, Netherlands

Abstract

Multi-hop question answering requires retrieving multiple pieces of evidence that jointly support an answer. However, retrieval-augmented QA systems often provide the answer generator with loosely selected passages or sentences, making it difficult to inspect and independently evaluate the evidence used for generation. We introduce **SAGE-QA**, a support-aware framework that separates evidence retrieval from downstream answer prediction by explicitly ranking candidate support sets before producing the final answer. SAGE-QA constructs question-conditioned evidence spaces, generates candidate support sets, and scores each candidate jointly using a graph neural network. A symbolic reranking stage then applies Text-Chain criteria for textual evidence and proof-aware criteria for ontology-based evidence. The selected support is then used for support-grounded answer prediction. We evaluate SAGE-QA on HotpotQA, 2WikiMultiHopQA, and controlled ontology-based datasets with explicit explanations. Results show that graph-based support ranking substantially improves evidence recall over lexical retrieval, while the additional reranking stage provides smaller and dataset-dependent gains. The analysis further reveals that retrieving complete evidence and producing compact support sets are distinct objectives: the system usually recovers the necessary evidence but often includes additional irrelevant units. These findings motivate support retrieval as an independently evaluable intermediate task and highlight support minimization as an important direction for reliable evidence-grounded QA.

Keywords

Neurosymbolic Question Answering, Multi-hop Question Answering, Retrieval-Augmented Generation, Evidence Retrieval, Ontology Reasoning

CEUR-WS.org/Vol-4312/paper18.pdf

1. Introduction

Multi-hop question answering (QA) requires a system to identify several pieces of evidence that jointly support an answer. The relevant information may be distributed across documents, sentences, knowledge-base assertions, or schema-level axioms, and individual evidence units may appear only weakly related to the question when considered in isolation. A successful multi-hop QA system must therefore retrieve not only individually relevant information, but also a compositionally sufficient support set that connects the entities, relations, and intermediate concepts required to derive the answer.

Retrieval-augmented generation (RAG) has become a widely used paradigm for grounding large language models (LLMs) in external knowledge [1, 2]. In a typical RAG pipeline, retrieved passages or sentences are serialized as context and passed directly to an answer generator. Although this can improve answer quality, the retrieved evidence is often treated as a flat collection of text fragments.

RAGE-KG 2026: The Third International Workshop on Retrieval-Augmented Generation Enabled by Knowledge Graphs, co-located with ISWC 2026, October 25–26, 2026, Bari, Italy

julie.loesch@maastrichtuniversity.nl (J. Loesch); michel.dumontier@maastrichtuniversity.nl (M. Dumontier); remzi.celebi@maastrichtuniversity.nl (R. Celebi)

0009-0009-0736-8389 (J. Loesch); 0000-0003-4727-9435 (M. Dumontier); 0000-0001-7769-4272 (R. Celebi)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-04

Consequently, it can be difficult to determine whether the selected context forms a coherent support chain, contains all evidence required to answer the question, or includes irrelevant information that may influence generation.

This distinction is particularly important in multi-hop QA. A correct answer does not necessarily imply that the appropriate supporting evidence was retrieved. An LLM may answer correctly from partial evidence, exploit dataset regularities, or rely on information that is not explicitly contained in the selected context. Conversely, a system may retrieve all necessary evidence but still fail to generate the correct answer. Evaluating answer quality alone therefore conflates two distinct capabilities: retrieving sufficient support and generating an answer from that support.

Benchmarks such as HotpotQA, 2WikiMultiHopQA, and MuSiQue provide sentence-level supporting-fact annotations, enabling evidence selection to be evaluated independently from answer generation [3, 4, 5]. In these settings, the supporting evidence often spans multiple documents and must be connected through bridge entities, shared relations, or comparison structures. Treating sentences as independent retrieval units may fail to capture these dependencies because the relevance of one sentence frequently depends on the presence of another.

A similar challenge arises in ontology-based QA. In this setting, a support set may contain instance-level assertions from the ABox, representing facts about specific entities, together with schema-level axioms from the TBox, representing general rules and relationships. Such axioms may encode domain and range restrictions, subclass relations, inverse properties, transitivity, symmetry, or property chains. The answer may therefore depend on a composition of facts and inference rules rather than on a single retrieved statement. An effective support-retrieval system must identify combinations of evidence units that jointly permit the queried statement to be derived.

Graph-based representations provide a natural way to model these dependencies. Evidence units can be connected through shared entities, relations, document titles, or fact–schema compatibility. Graph neural networks (GNNs) can then propagate information across these connections and score evidence structures using both semantic and relational signals. However, neural ranking alone does not guarantee that a selected support is compact, compositionally sufficient, or compatible with domain-specific reasoning requirements. It may retrieve the required evidence together with redundant or irrelevant units.

Symbolic criteria can complement neural ranking by explicitly encoding properties such as query coverage, connectivity, proof feasibility, and support compactness. Nevertheless, the contribution of such criteria must be evaluated separately from the contribution of graph-based neural ranking. In particular, it is necessary to determine whether symbolic reranking provides measurable benefits beyond a learned GNN ranker and whether those benefits concern evidence recall, evidence precision, answer quality, or explanation minimality.

We introduce **SAGE-QA**, a support-aware framework that separates structured evidence retrieval from downstream answer prediction. Rather than ranking individual evidence units independently, SAGE-QA constructs a bounded candidate-support space and scores each candidate set jointly using a GNN-based support ranker. The initial ranking is subsequently refined using Text-Chain criteria for natural-language evidence or proof-aware criteria for ontology-based evidence. The highest-ranked candidates are then aggregated and used as grounded evidence for downstream answer prediction.

For text-based QA, the evidence units are sentences from the provided context. Auxiliary knowledge-graph triples are used to construct relational connections between sentences, entities, and document titles, while supervision and evaluation remain sentence-based. For ontology-based QA, evidence units consist of factual assertions and schema axioms, and candidate supports correspond to combinations

that may form a valid symbolic justification. This shared formulation makes it possible to study support retrieval across two complementary settings: natural-language benchmarks with ground-truth supporting facts and controlled ontology-based datasets with explicit explanation sets.

Figure 1 illustrates the central distinction between complete and compact support, complete but non-minimal support, and incomplete support in both the textual and ontology-based settings.

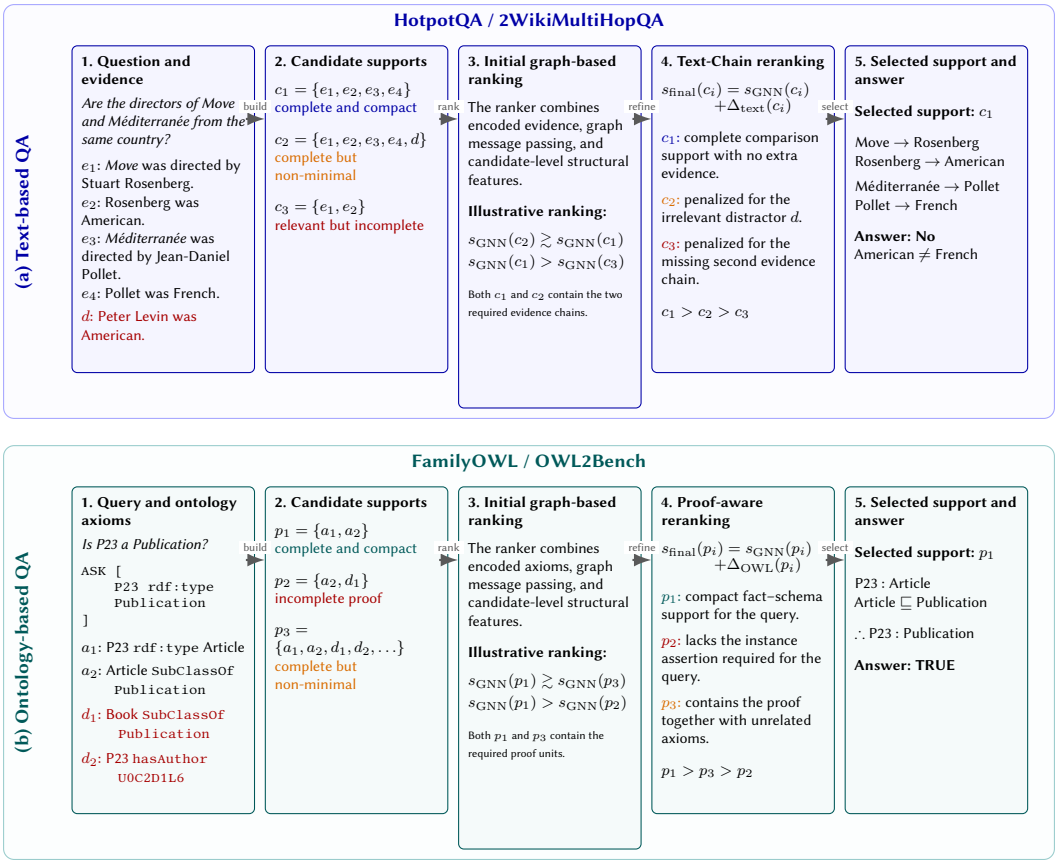


Figure 1: Walk-through of SAGE-QA for text-based and ontology-based QA. Candidate support sets are first scored jointly by the graph-based ranker and then refined using Text-Chain or proof-aware reranking. The rankings are illustrative. For clarity, the walk-through shows the top-ranked candidate as the selected support; in the experiments, the top-k candidates are aggregated as defined in Equation 4.

The main contributions of this paper are:

1. A unified candidate-level support-retrieval formulation for text-based and ontology-based QA, in which candidate support sets are scored jointly rather than individual evidence units independently.
2. A support-ranking framework that combines graph-based candidate scoring with domain-specific symbolic reranking, using Text-Chain criteria for textual evidence and proof-aware criteria for ontology-based evidence.
3. An evaluation framework that separately measures retrieval quality, final support quality, answer quality, and joint performance across text-based and ontology-based QA.

2. Related Work

2.1. Retrieval-Augmented and Multi-Hop QA

Retrieval-augmented generation grounds answer generation in information retrieved from an external corpus or knowledge source [1, 2]. Multi-hop QA extends this setting because the required evidence is distributed across several sources. Earlier systems retrieve iteratively using entities or representations from previously selected passages [6, 7, 8]; more recent methods interleave retrieval with intermediate reasoning, including IRCoT [9], TRACE [10], and HopRAG [11]. SAGE-QA addresses a related but distinct task: it constructs and ranks complete candidate support sets before downstream answer prediction, allowing support quality to be evaluated independently from the final answer.

2.2. Graph-Based and Joint Evidence Retrieval

Graphs have been used to represent documents, sentences, entities, and candidate answers in multi-hop QA. CogQA, DGN, DFGN, and HGN apply graph-based reasoning or message passing to retrieve supporting facts and predict answers [12, 13, 14, 15]. At the same time, prior work shows that strong contextual encoders can sometimes match graph-based components [16, 17], motivating explicit comparison with a GNN-only ablation. Joint evidence-set retrieval also predates this work: Yadav et al. [18] score linked sentence clusters rather than independent sentences, while other approaches represent intermediate reasoning as evidence chains or graphs [19, 10]. SAGE-QA builds on these ideas by applying a shared candidate-support formulation to both textual and ontology-based evidence and by evaluating symbolic reranking as an additional component beyond graph-based candidate scoring.

2.3. Knowledge-Graph and Ontology-Based QA

Graph retrieval has also been applied directly to knowledge graphs. GNN-RAG retrieves answer candidates over a dense KG subgraph and extracts shortest paths connecting question entities to predicted answers [20], while KG-RAR alternates graph retrieval and reasoning [21]. Neurosymbolic systems additionally combine language models with explicit structured reasoning; for example, ORACLE constructs question-specific ontologies and decomposes multi-hop questions into executable sub-questions [22]. In contrast, SAGE-QA uses symbolic information specifically for support selection: instance-level facts and schema axioms are treated as retrievable evidence units, and proof-related criteria refine a bounded candidate-support space rather than replacing neural retrieval with unrestricted symbolic inference.

3. Methodology

SAGE-QA separates two components that are often evaluated together in question answering: retrieving the evidence needed to support an answer and predicting the answer from that evidence. Given a question, the framework first constructs several plausible candidate support sets, ranks each set as a whole, and then refines the ranking using domain-specific structural criteria. Only after this retrieval stage is complete is the selected evidence used for downstream answer prediction. This separation makes it possible to determine whether an error originates from missing or irrelevant evidence, or from the downstream answer-prediction stage.

The same retrieval abstraction is used for text-based and ontology-based QA. In text-based QA, an evidence unit is a sentence, while auxiliary relational triples provide additional graph structure but are not themselves returned or evaluated as support. In ontology-based QA, an evidence unit is either an instance-level fact or a schema-level axiom.

Throughout this paper, we use *ground-truth support* to denote the reference evidence associated with a question: the benchmark-provided supporting facts for text-based QA and the reasoner-generated explanation sets for ontology-based QA.

3.1. Evidence and Candidate Construction

Let q denote a question and let $\mathcal{U}_q = \{u_1, \ldots, u_n\}$ denote the bounded pool of evidence units from which candidate supports are constructed. For text-based QA, an evidence unit is a sentence from the provided context; for ontology-based QA, an evidence unit is an instance-level fact or schema-level axiom from the local ontology context. Each question has one or more ground-truth support sets $\mathcal{E}(q) = \{E_1, \ldots, E_m\}$, with $E_j \subseteq \mathcal{U}_q$. SAGE-QA constructs a set of candidate supports

$$\mathcal{C}(q) = \{C_1, \ldots, C_K\}, \qquad C_i \subseteq \mathcal{U}_q, \tag{1}$$

and scores each candidate jointly as a complete set rather than scoring its evidence units independently. A candidate is *complete* if it contains every evidence unit of at least one ground-truth support set, i.e., $E_j \subseteq C_i$ for some $E_j \in \mathcal{E}(q)$. It is *exact* if $C_i = E_j$. A complete but non-exact candidate therefore contains all required evidence together with one or more additional units.

Text-based QA. For HotpotQA and 2WikiMultiHopQA, the provided context is flattened into sentence-level evidence units identified by document title, sentence index, and sentence text. A bounded pool of at most 30 sentences is constructed using lexical overlap between the question and the sentence and document title. Candidate supports are formed from single sentences and sentence combinations of up to three units for HotpotQA and four units for 2WikiMultiHopQA, with at most 256 candidates retained per question.

To introduce relational structure, the evidence graph is augmented with auxiliary subject–predicate–object triples. For HotpotQA, these triples are extracted from the textual context using `GPT-4.1-mini`; for 2WikiMultiHopQA, dataset-provided triples are preserved and augmented with LLM-extracted triples.

During construction of the supervised candidate-ranking data, ground-truth support units are explicitly retained to ensure that positive candidates are available. We also include supersets of the ground-truth support so that the ranker is exposed to complete but non-minimal alternatives. Ground-truth support is used only to construct the supervision targets and is not included in the neural or symbolic scoring features.

Accordingly, the reported retrieval results evaluate ranking within a bounded candidate-support space rather than unconstrained retrieval from the full source corpus.

Ontology-based QA. Each ontology-based example consists of a natural-language question, its corresponding SPARQL query, and an OWL context. The SPARQL query is used to identify query-relevant entities and properties and to guide candidate construction and ranking. Thus, the ontology

experiments assume that the structured query is available and do not evaluate the generation of SPARQL queries from natural language.

The OWL context corresponds to either a 1-hop or 2-hop neighborhood. A 1-hop context contains triples directly connected to the anchor individual, whereas a 2-hop context additionally contains triples reachable within two hops. Relevant schema axioms are included in both settings.

Entities and properties extracted from the SPARQL query are used to identify query-relevant ontology units. At most 40 context units are retained, with ground-truth support units additionally preserved for supervised candidate construction. Units are connected when they share an entity or property, and candidate supports are generated using connected beam search with beam width 96 and a maximum of 320 additional candidate subgraphs per question. Rather than enumerating every possible combination of ontology units, which would be computationally expensive, the beam search incrementally expands a limited number of promising connected candidates. The search favors query-relevant units, connected candidates, and combinations containing both factual and schema-level evidence. No fixed maximum support size is imposed; search is instead bounded by the context size, beam width, and candidate budget.

3.2. Graph-Based Evidence Ranking

Each candidate is ranked using the question-conditioned evidence graph $\mathcal{G}_q$. Its nodes include the available support units and, for text-based QA, the auxiliary relational triples. The graph is represented with bidirectional edges and self-loops.

The question and graph nodes are encoded into dense representations, after which a GraphSAGE-based encoder propagates information over $\mathcal{G}_q$. Each node representation is augmented with an eight-dimensional structural feature vector encoding whether the unit is a fact or rule, its overlap with query entities and properties, rule–query-property and fact–query-entity matches, normalized graph degree, and a bias feature. The representation of candidate C_i is obtained by pooling the contextualized embeddings of its support units. Auxiliary triples may influence this representation through message passing but are not included in the pooled support.

The final score combines the question representation, the pooled candidate representation, and structural features:

$$\mathbf{z}_i = \mathrm{MLP}_{\psi}(\boldsymbol{\psi}(q, C_i)), \qquad s_i^{\mathrm{GNN}} = \sigma(\mathrm{MLP}_{\mathrm{score}}(\mathbf{q} \| \mathbf{g}_i \| \mathbf{z}_i)), \tag{2}$$

The resulting scores define the initial candidate ordering.

Here, $\mathbf{q}$ is the encoded natural-language question representation, $\mathbf{g}_i$ is the mean-pooled representation of the contextualized graph nodes belonging to candidate C_i, $\boldsymbol{\psi}(q, C_i)$ is an eight-dimensional label-free candidate-level feature vector, and $\mathbf{z}_i$ is its learned projection. The operator $\|$ denotes concatenation and σ denotes the sigmoid function. Graph nodes are contextualized using two GraphSAGE layers before candidate pooling. Intuitively, message passing allows the relevance of an evidence unit to depend on the evidence to which it is connected. A sentence or ontology fact that only weakly matches the question in isolation may therefore receive a more informative representation when it connects other units that are jointly needed to support the answer.

For text-based QA, $\boldsymbol{\psi}$ summarizes question–candidate lexical alignment, document-title coverage, document diversity, cross-document structure, candidate size, and sentence position. For ontology-based QA, it summarizes overlap with query entities and properties, fact–schema composition, and candidate size. All candidate-level features used for ranking are label-free.

Fact–schema composition is a binary indicator of whether a candidate contains at least one factual assertion and at least one schema-level axiom.

3.3. Symbolic Reranking

The graph-based ranker may assign similar scores to candidates that contain the same core evidence but differ in completeness or redundancy. SAGE-QA therefore refines the initial score as

$$s_i^{\mathrm{SAGE}} = s_i^{\mathrm{GNN}} + \begin{cases} \Delta_{\mathrm{text}}(q, C_i), & \text{for text-based QA,} \\ \Delta_{\mathrm{OWL}}(q, C_i), & \text{for ontology-based QA.} \end{cases} \tag{3}$$

The two adjustments use fixed, label-free structural criteria defined below. Their coefficients are fixed heuristic weights rather than learned parameters and are applied consistently across all reported experiments; the exact values are reported in Appendix B.2.

Because s_i^{GNN} is bounded to $[0, 1]$ by the sigmoid function, the symbolic terms act as comparatively small bonuses and penalties that refine the neural ordering rather than replace it. The symbolic criteria are therefore evaluated as reranking components, with the GNN Subgraph variant providing the corresponding GNN-only ablation.

Text-Chain reranking. Text-Chain reranking uses only label-free structural signals. It rewards (i) coverage of question terms by document titles, (ii) lexical overlap between the question and candidate support, (iii) evidence spanning multiple documents, (iv) bridge connectivity, measured by whether the title or entity of one support sentence occurs in another support sentence, (v) connectivity between candidate sentences and auxiliary KG triples, and (vi) multi-entity coverage for comparison questions. Candidates are penalized for excessive support size and for multi-sentence supports drawn from only one document.

Proof-aware reranking. For binary ontology questions, proof-aware reranking uses a deterministic OWL proof module to test whether a candidate support entails the target ASK query. Candidates for which the queried statement can be derived receive a proof bonus. For open-ended SELECT questions, this proof bonus is not applied, and reranking relies on query coverage, fact–schema composition, and compactness. Oversized supports and candidates containing unnecessary schema axioms are penalized. All reranking signals are computed without access to the ground-truth support.

3.4. Support-Grounded Answer Prediction

The final support is formed from the retrieved support:

$$\widehat{S}_k(q) = \bigcup_{C_i \in \mathrm{TopK}(q)} C_i. \tag{4}$$

The final support is provided to the downstream answer-prediction stage together with the question. For text-based QA, only the selected sentence units are included in the reader context and are passed to `GPT-4.1-mini`. Auxiliary KG triples may affect ranking through graph connectivity and message passing, but they are excluded from the returned support set, are not evaluated as supporting facts, and are not passed to the reader.

For ontology-based QA, the final context consists of the selected instance-level facts and schema-level axioms. For binary ASK questions, a deterministic OWL proof procedure first tests whether the retrieved support entails the queried statement. If a derivation is found, the entailed answer is returned directly. Otherwise, the retrieved support is passed to `GPT-4.1-mini`. Open-ended SELECT questions are answered with `GPT-4.1-mini`.

4. Experiments

4.1. Datasets

Text-based QA. We use *HotpotQA* and *2WikiMultiHopQA* [3, 4], with 3,000/500/1,000 train/development/evaluation examples per dataset. Sentences are the support units; auxiliary relational triples provide graph structure. HotpotQA triples are LLM-extracted, while 2WikiMultiHopQA preserves dataset-provided triples and augments them with LLM-extracted triples.

Ontology-based QA. We use CORE-LLM-Bench datasets derived from the *Family* [23] and *OWL2Bench* [24] ontologies in 1-hop and 2-hop settings. Each question is paired with explicit supporting facts and schema axioms; generation details are provided in Appendix B.1.

4.2. Models and Baselines

The internal methods—Lexical Subgraph, GNN Subgraph, and SAGE-QA—operate over the same dataset-specific candidate support space. Candidate support sets are ranked, and the deduplicated union of the evidence units contained in the top-ranked candidates forms the retrieved support used for downstream answer prediction. GNN-RAG retains its native graph-retrieval and answer-prediction pipeline and is included as an external system-level baseline.

Here, k denotes the number of candidate support sets rather than the number of individual evidence units. Consequently, the final retrieved support has variable size depending on the size and overlap of the selected candidates; no additional fixed limit is imposed on the number of unique evidence units in their union.

We fix $k = 3$ as a small high-recall aggregation budget for the primary experiments. The motivation is to reduce dependence on a single ranking decision: if the top-ranked candidate omits one required piece of evidence, aggregating a small number of highly ranked candidates allows complementary evidence to be recovered. At the same time, increasing k introduces additional evidence and therefore reduces support compactness. We use $k = 3$ as a fixed compromise rather than selecting it by maximizing test-set F1. Appendix B.4 evaluates $k \in \{1, 2, 3, 5\}$ and quantifies this trade-off.

We compare five variants. *Lexical Subgraph* ranks the candidate sets by question–candidate lexical overlap. *GNN Subgraph* is the GNN-only ablation. *SAGE-QA Text-Chain* and *SAGE-QA Proof* add the text- and ontology-specific reranking criteria described in Section 3.3. *GNN-RAG* [20] is an external graph-neural RAG baseline that retrieves KG reasoning paths and uses its native `GPT-4.1-mini`-based answer module; adaptation details are given in Appendix B.6.

4.3. Evaluation Metrics

We evaluate SAGE-QA using both retrieval-only support and end-to-end QA metrics.

Support Retrieval metrics. For each question, the retriever returns the top k candidate support sets. Retrieval is evaluated over the deduplicated union of the evidence units contained in these candidates. Precision@k is the fraction of retrieved evidence units that belong to the ground-truth support, while Recall@k is the fraction of ground-truth support units recovered in the retrieved support. F1@k is their harmonic mean, balancing evidence completeness and compactness.

For example, if the retrieved union contains both required support units and one additional unit, Precision@$3 = 2/3$, Recall@$3 = 1$, and F1@$3 = 0.8$.

End-to-end QA metrics. For all datasets, we evaluate answer quality with exact match *Answer EM* and token-level F1 *Answer F1*, and support quality with exact match *Support EM* and F1 *Support F1* over the retrieved top-k context used by the downstream answer-prediction stage. Because these metrics are computed on the final retrieved context, they penalize cases where the ground-truth support is retrieved together with irrelevant evidence. For HotpotQA and 2WikiMultiHopQA, support units are title–sentence-index pairs following Hotpot-style supporting-fact evaluation [3]; for ontology-based datasets, they are facts and schema-level axioms from the ground-truth support sets. We also report HotpotQA-style joint metrics: *Joint EM* requires both answer and support exact matches, while *Joint F1* combines answer- and support-level scores.

4.4. Experimental Setup

All SAGE-QA GNN variants use `google/bert_uncased_L-2_H-128_A-2` as the text encoder, followed by two GraphSAGE layers with hidden dimension 128 and dropout 0.1. The encoder is frozen in the reported experiments. Models are trained for three epochs using AdamW with a learning rate of 2×10^{-5}, maximum sequence length 128, candidate batch size 512, and random seed 42.

Training uses a within-question pairwise ranking objective with margin 0.2 and weight 2.0, together with an auxiliary binary cross-entropy loss with weight 0.2. Exact support sets receive a ranking target of 1.0, complete support sets 0.9, partially overlapping supports at most 0.3, and irrelevant supports 0.0. At most 512 ranking pairs are sampled per question. The best checkpoint is selected according to top-1 support F1 on the development set.

For text-based QA, the internal methods use `GPT-4.1-mini` as a fixed reader. Keeping the downstream reader constant allows differences among these methods to be attributed primarily to support retrieval rather than to the choice of language model. For ontology-based QA, the internal methods use the proof-first answer-prediction procedure described in Section 3.4: binary queries are answered directly when the retrieved support yields a deterministic proof, and `GPT-4.1-mini` is used otherwise and for open-ended questions. GNN-RAG retains its native `GPT-4.1-mini`-based answer-prediction module.

5. Results

5.1. Support Retrieval Performance

Table 1 reports retrieval performance the proof-first answer-prediction procedure. The metrics are computed over the union of evidence units contained in the top-3 ranked candidate support sets and therefore evaluate the final retrieval context independently of the downstream answer stage.

Table 1
Retrieval-only support results the proof-first answer-prediction procedure. Bold indicates best. GNN-RAG is omitted because its retrieved paths are not directly comparable to SAGE-QA candidate support sets.

Dataset	Method	Precision@3	Recall@3	F1@3
Text-based QA				
HotpotQA	Lexical Subgraph	0.465	0.844	0.585
	GNN Subgraph	0.472	0.944	0.615
	SAGE-QA Text-Chain	**0.477**	**0.945**	**0.618**
2WikiMultiHopQA	Lexical Subgraph	0.424	0.815	0.546
	GNN Subgraph	0.531	**0.992**	0.665
	SAGE-QA Text-Chain	**0.533**	**0.992**	**0.667**
Ontology-based QA				
Family 1-hop	Lexical Subgraph	0.523	0.781	0.588
	GNN Subgraph	**0.625**	0.971	**0.711**
	SAGE-QA Proof	0.613	**0.994**	0.710
Family 2-hop	Lexical Subgraph	0.451	0.914	0.502
	GNN Subgraph	**0.512**	0.973	**0.563**
	SAGE-QA Proof	0.494	**0.996**	0.556
OWL2Bench 1-hop	Lexical Subgraph	0.483	0.978	0.583
	GNN Subgraph	**0.507**	0.978	**0.599**
	SAGE-QA Proof	0.497	**1.000**	0.597
OWL2Bench 2-hop	Lexical Subgraph	0.460	0.974	0.529
	GNN Subgraph	0.481	0.983	0.546
	SAGE-QA Proof	**0.488**	**1.000**	**0.561**

Text-based QA. Graph-based ranking substantially improves support recovery over lexical retrieval on both datasets. SAGE-QA Text-Chain gives the best retrieval scores, but its gains over GNN Subgraph are small, indicating that most of the improvement comes from candidate-level graph ranking.

Ontology-based QA. SAGE-QA Proof achieves the highest Recall@3 in all four settings, including perfect recall on both OWL2Bench splits, whereas GNN Subgraph sometimes yields higher precision and F1. Proof-aware reranking therefore favors complete proof recovery at the cost of additional axioms, particularly in the 2-hop settings.

5.2. End-to-End QA

Table 2 reports answer, support, and joint performance after aggregating the retrieved top-3 candidate support sets and applying the corresponding support-grounded answer-prediction stage.

Text-based QA. SAGE-QA achieves the best or near-best answer and joint performance on HotpotQA, whereas GNN Subgraph is slightly stronger on 2WikiMultiHopQA. Thus, better support retrieval does not always translate directly into better answers, and strict Support EM remains low when complete support is retrieved together with additional sentences.

Ontology-based QA. Proof-aware reranking improves Answer EM and Answer F1 over GNN Subgraph in all four settings, with the clearest gains on OWL2Bench. Support and joint metrics are more mixed because complete proofs are often retrieved together with additional axioms, reflecting the same completeness–compactness trade-off observed in retrieval.

Table 2
End-to-end support-grounded QA performance. The internal text-based methods use `GPT-4.1-mini` for answer generation. For the internal ontology-based methods, binary queries use deterministic proof when possible and `GPT-4.1-mini` otherwise; GNN-RAG retains its native answer-prediction pipeline. A = answer, S = support, and J = joint. Bold indicates best.

Dataset	Method	A-EM	A-F1	S-EM	S-F1	J-EM	J-F1
Text-based QA							
HotpotQA	Lexical Subgraph	0.615	**0.779**	0.026	0.585	**0.018**	0.471
	GNN-RAG	0.472	0.674	0.019	**0.632**	0.011	0.432
	GNN Subgraph	0.619	0.778	0.024	0.615	0.012	0.491
	SAGE-QA Text-Chain	**0.620**	**0.779**	**0.029**	0.618	0.015	**0.493**
2WikiMultiHopQA	Lexical Subgraph	0.616	0.679	0.029	0.546	0.027	0.397
	GNN-RAG	0.513	0.670	**0.557**	**0.853**	**0.304**	**0.570**
	GNN Subgraph	**0.705**	**0.767**	0.197	0.665	0.171	0.521
	SAGE-QA Text-Chain	0.701	0.765	0.197	0.667	0.172	0.522
Ontology-based QA							
Family 1-hop	Lexical Subgraph	0.794	0.810	0.303	0.588	0.296	0.540
	GNN-RAG	0.853	0.858	**0.373**	0.709	**0.350**	0.628
	GNN Subgraph	0.918	0.937	0.354	**0.711**	0.344	0.684
	SAGE-QA Proof	**0.926**	**0.942**	0.344	0.710	0.338	**0.689**
Family 2-hop	Lexical Subgraph	0.872	0.894	0.296	0.502	0.290	0.467
	GNN-RAG	0.732	0.737	0.296	0.518	0.252	0.406
	GNN Subgraph	0.916	0.932	**0.334**	**0.563**	**0.322**	**0.530**
	SAGE-QA Proof	**0.922**	**0.940**	0.328	0.556	0.315	0.529
OWL2Bench 1-hop	Lexical Subgraph	0.782	0.851	**0.264**	0.583	0.240	0.536
	GNN-RAG	0.497	0.538	0.152	0.364	0.072	0.238
	GNN Subgraph	0.769	0.847	**0.264**	**0.599**	**0.244**	0.545
	SAGE-QA Proof	**0.809**	**0.876**	0.260	0.597	**0.244**	**0.569**
OWL2Bench 2-hop	Lexical Subgraph	0.850	0.920	0.310	0.529	0.295	0.484
	GNN-RAG	0.654	0.692	0.202	0.412	0.120	0.272
	GNN Subgraph	0.860	0.922	0.310	0.546	**0.298**	0.498
	SAGE-QA Proof	**0.875**	**0.940**	**0.314**	**0.561**	0.295	**0.523**

6. Discussion

6.1. Effect of Symbolic Reranking

Comparing SAGE-QA with the GNN Subgraph ablation shows that symbolic reranking mainly refines rather than replaces the learned ranking. Text-Chain reranking provides small but consistent retrieval gains by distinguishing candidates that differ in cross-document coverage or redundancy. Proof-aware reranking affects recall more consistently than precision, promoting complete fact–schema combinations while occasionally retaining non-minimal supersets. Its impact is therefore greatest when semantically similar candidates differ in compositional sufficiency.

6.2. Support Quality and Error Analysis

The detailed error analysis reveals three main patterns: missed evidence, over-retrieval, and downstream answer failure despite sufficient retrieved evidence.

Missed evidence. Complete retrieval failure is uncommon. Only 3.3% of HotpotQA examples miss the ground-truth support entirely, while no such failures are observed for 2WikiMultiHopQA in the error analysis. The ontology datasets exhibit the same pattern, with near-zero average missing support across all settings.

Over-retrieval. The dominant error type is over-retrieval, where the complete ground-truth support is recovered together with additional evidence. This pattern is consistent with aggregating the union of the top-3 candidate supports, which prioritizes recall but may retain evidence from several overlapping or non-minimal candidates. Consequently, the final retrieved support can be larger than the ground-truth support, allowing answer accuracy to remain high while support exact match and precision decrease.

Downstream answer failure. A second important failure mode occurs when the required evidence is present but the downstream answer-prediction stage still returns an incorrect answer. This shows that retrieval completeness is necessary but not sufficient: the selected evidence must also be used correctly to produce the final answer.

These findings identify support compression as an important remaining challenge. Future work should preserve the high recall of candidate aggregation while selecting a smaller subset of non-redundant evidence units before downstream answer prediction. A detailed error analysis is provided in Appendix B.5.

6.3. Limitations

The evaluation has several limitations. First, SAGE-QA is evaluated as a bounded candidate-ranking problem in which ground-truth support units are retained during supervised candidate construction; the reported results therefore do not measure unconstrained retrieval from a full corpus or ontology. Second, the ontology-based setting assumes access to the structured SPARQL query in addition to the natural-language question. Third, the text-based evidence graphs partly rely on LLM-extracted auxiliary triples. These triples are used only to provide graph structure and are neither returned as support nor passed to the downstream reader; however, their extraction accuracy is not evaluated separately, and extraction errors may therefore affect graph connectivity and ranking. Finally, the improvements from symbolic reranking over the GNN-only ablation are generally small, and no statistical significance tests are reported. These differences should therefore be interpreted descriptively rather than as evidence of large performance gains.

7. Conclusion

We presented SAGE-QA, a neurosymbolic framework for support-grounded multi-hop question answering. SAGE-QA constructs question-conditioned candidate support sets, ranks them with a graph-based retriever, and refines the ranking using Text-Chain or proof-aware reranking before downstream answer prediction.

Experiments on text-based and ontology-based QA show that candidate-level graph ranking substantially improves support recovery over lexical retrieval, while symbolic reranking further refines evidence coverage and compositional sufficiency. The results also show that high support recall does not necessarily yield minimal or exact support. Aggregating evidence from the top-3 candidates often retrieves the required evidence but may introduce redundant or irrelevant units, reducing support precision and exact match.

Future work should therefore focus on support compression that preserves complete evidence while removing non-essential units before downstream answer prediction.

A. Source Code

The implementation used for the experiments is publicly available at: https://github.com/jloe2911/SAGE-QA.

B. Supporting Material

B.1. CORE-LLM-Bench

This section describes how CORE-LLM-Bench is used to generate the controlled ontology-based datasets used in this paper. The goal is to create question-answer pairs with explicit symbolic explanations, enabling direct evaluation of support retrieval over instance-level facts and schema-level axioms. The overall pipeline is summarized in Table 3.

The data creation procedure consists of four stages:

1. **Ontology slicing:** For each anchor individual, we extract a local ontology subgraph containing triples in which the individual appears as either subject or object. We construct both 1-hop neighborhoods, containing triples directly connected to the anchor resource, and 2-hop neighborhoods, containing triples reachable within two hops. Relevant schema axioms are included to preserve the reasoning context.
2. **Symbolic inference and query generation:** A symbolic reasoner is applied to each local subgraph to derive entailed triples. These inferred triples are used as target answers, and corresponding SPARQL queries are generated in both binary and open-ended forms. Binary queries are represented as positive and negative ASK queries, while open-ended queries are represented as SELECT queries.
3. **Metadata assignment:** Each generated query is assigned metadata describing ontology size and reasoning complexity. Ontology size is measured as the total number of triples in the corresponding 1-hop or 2-hop individual-centric graph. Reasoning complexity is derived from Pellet-generated explanations using a tagging framework with 20 reasoning tags. Each explanation path is encoded as a tag string, where repeated reasoning types are represented by repeated tags, such as HH for two hierarchy steps or DHT for direct assertion, hierarchy, and transitivity. For each query, the longest valid explanation tag sequence is used as its reasoning-complexity value.
4. **Stratified sampling:** The metadata is used to sample a balanced set of question-answer pairs across ontology sizes and reasoning-complexity levels. This produces datasets that cover both shallow and more compositionally demanding OWL reasoning cases.

The outcome of this sampling process is summarized in Table 4, which reports the generated and sampled question counts for each ontology and hop setting.

B.2. Symbolic Reranking Details

For ontology-based QA, the proof-aware adjustment is

$$\Delta_{\mathrm{OWL}} = 0.180P + 0.030Q + 0.020F + 0.350\Delta_{\mathrm{compact}} - 0.010\max(0, |C_i| - 2), \tag{5}$$

Table 3
Overview of the CORE-LLM-Bench data creation pipeline using a running example centered around John. Individual-centric ontology subgraphs are processed by a symbolic reasoner to obtain entailed inferences. These inferences are transformed into positive and negative ASK queries for Binary Question Answering (BQA) and SELECT queries for Open-Ended Question Answering (OEQA). The generated queries are then assigned metadata and sampled to produce balanced controlled ontology-based datasets.

Pipeline step	Running example
1. Ontology slicing	An individual-centric subgraph is extracted around John. **ABox:** John rdf:type Person; John hasParent Mary; John hasSibling Lisa. **TBox:** Woman $\sqsubseteq$ Person; hasParent inverseOf hasChild; hasSibling symmetric.
2. Symbolic inference and query generation	Pellet is applied to the local subgraph to derive entailed inferences, which serve as benchmark answers. **Inferences:** Mary hasChild John; Lisa hasSibling John. Task-specific SPARQL queries are then generated from each inferred triple. **BQA:** Positive ASK queries test entailed triples, e.g., ASK WHERE { Mary hasChild John . }. Matched negative ASK queries preserve the subject and predicate but replace the object with a non-entailed term, e.g., ASK WHERE { Mary hasChild Lisa . }. **OEQA:** SELECT queries retrieve the inferred entity, class, or property value, e.g., SELECT ?x WHERE { Mary hasChild ?x . }.
3. Metadata assignment	Each generated query is assigned metadata based on ontology size and reasoning complexity. **Size:** Ontology size is measured as the total number of triples in the corresponding 1-hop or 2-hop individual-centric graph. **Reasoning complexity:** Pellet-generated explanations are categorized using a tagging framework with 20 reasoning tags. Each explanation path is encoded as a tag string, where repeated reasoning types are represented by repeated tags, e.g., HH for two hierarchy steps or DHT for direct assertion, hierarchy, and transitivity. For each query, the longest valid explanation tag sequence is used as its reasoning-complexity value.
4. Stratified sampling	The assigned metadata is used to sample a balanced set of question-answer pairs across ontology sizes and reasoning-complexity levels. The resulting datasets cover both shallow and more compositionally demanding OWL reasoning cases.

Table 4
Statistics for the generated and sampled benchmark datasets used in the experiments. For each ontology, we report the number of binary questions (BQ), open-ended questions (OEQ), sampled binary questions, sampled open-ended questions, and total sampled questions, separately for the 1-hop and 2-hop settings. Values in parentheses indicate positive/negative binary-question counts.

Ontology	Hop	BQ	OEQ	Sampled BQ	Sampled OEQ	Sampled Questions
Family	1-hop	5012 (2506/2506)	2506	1272 (603/669)	608	1880
	2-hop	5012 (2506/2506)	2506	1272 (602/670)	608	1880
OWL2Bench	1-hop	3912 (1956/1956)	1956	957 (484/473)	509	1466
	2-hop	4246 (2123/2123)	2123	1044 (508/536)	546	1590

where $P = 1$ when the deterministic proof module derives a binary ASK query and $P = 0$ otherwise, Q measures query coverage, and F indicates that the candidate contains both factual and schema-level evidence. The compactness component is

$$\Delta_{\text{compact}} = 0.025R_p + 0.015R_e + 0.020F - 0.006\max(0, |C_i| - 2) - 0.004\max(0, N_s - 1), \quad (6)$$

where R_p denotes query-property match, R_e query-entity coverage, and N_s the number of schema axioms in the candidate.

For text-based QA, the reranking adjustment is

$$\Delta_{\text{text}} = 0.030T + 0.020Q + 0.020D + 0.025B + 0.030K + 0.020C - P_{\text{size}} - P_{\text{dup}}, \tag{7}$$

where T is question–title coverage, Q is question–candidate overlap, D is cross-document coverage, B is bridge connectivity, K is candidate–KG connectivity, and C is comparison-question coverage. $P_{\text{dup}} = 0.010$ for multi-sentence candidates containing evidence from only one document and 0 otherwise. The size penalty is 0 for candidates of size at most two, 0.002 for size three, 0.004 for size four, and $0.010(|C_i| - 4)$ for larger candidates.

B.3. LLM Answer-Generation Prompts

All LLM-based answer-prediction calls use `GPT-4.1-mini`. For the internal methods, the same reader prompt is used across retrieval variants within each benchmark family, so that Lexical Subgraph, GNN Subgraph, and SAGE-QA differ only in the retrieved evidence supplied to the downstream answer stage.

Text-based QA. For HotpotQA and 2WikiMultiHopQA, the system message is:

```
You are a careful question answering system. Use only the given evidence.
```

The user prompt is:

```
You are answering a multi-hop question using retrieved reasoning paths.

Reasoning Paths:
[RETRIEVED EVIDENCE]

Question:
[QUESTION]

Answer Type Guidance:
[QUESTION-TYPE GUIDANCE]

Instructions:
- Use only the reasoning paths above. Do not use outside knowledge.
- Return the shortest possible answer.
- For yes/no questions, answer exactly "yes" or "no".
- For multiple answers, separate items with semicolons.
- If the reasoning paths are insufficient, answer "unknown".
- Provide a one-sentence explanation grounded in the reasoning paths.

Return valid JSON only, with exactly these keys:
{
  "answer": "...",
  "explanation": "..."
}
```

Retrieved sentences are serialized with their document title and sentence index. Question-type guidance specifies the expected answer form for comparison, entity, place, and Boolean questions. The model is queried with temperature 0.

Table 5
Sensitivity to the number of aggregated candidate supports. Each cell reports Precision / Recall / F1.

Dataset	$k = 1$	$k = 2$	$k = 3$	$k = 5$
HotpotQA	.734/.924/.802	.570/.935/.693	.477/.945/.618	.356/.958/.506
2WikiMultiHopQA	.721/.975/.822	.607/.989/.735	.533/.992/.667	.436/.993/.568
Family 1-hop	.947/.990/.958	.680/.992/.768	.613/.994/.710	.543/1.000/.646
Family 2-hop	.998/.991/.992	.531/.997/.589	.494/.996/.556	.461/.998/.528
OWL2Bench 1-hop	.984/.970/.973	.583/1.000/.676	.497/1.000/.597	.433/1.000/.525
OWL2Bench 2-hop	.994/.945/.960	.545/.984/.601	.488/1.000/.561	.440/1.000/.509

Ontology-based QA. For FamilyOWL and OWL2Bench, a deterministic proof procedure is first applied to supported binary queries. When no answer is derived, `GPT-4.1-mini` receives the following system message:

```
You answer questions using only provided evidence and return valid JSON.
```

The user prompt is:

```
You are answering an ontology-grounded question using retrieved reasoning paths.

Reasoning Paths:
[RETRIEVED EVIDENCE]

Question:
[QUESTION]

Instructions:
- Answer using only the retrieved reasoning paths.
- Treat Facts as instance-level evidence and Axioms as schema/rule evidence.
- If a fact and the necessary axiom together entail the question, answer with the
    entailed result.
- If the reasoning paths are insufficient, answer "Unknown"; do not guess.
- [QUESTION-TYPE ANSWER FORMAT]
- Return valid JSON only, with this exact schema:
{"answer": "...", "explanation": "..."}
```

For binary questions, the required answer format is `TRUE`, `FALSE`, or `Unknown`; for open-ended questions, the model is instructed to return the shortest entity, name, or value that answers the question. Retrieved ontology units are labeled as facts or axioms before being supplied to the reader.

B.4. Sensitivity to the Number of Retrieved Candidates

Table 5 reports retrieval performance when aggregating the top $k \in \{1, 2, 3, 5\}$ SAGE-QA candidates. The metrics are computed over the deduplicated union of the selected candidate supports, using the same evaluation procedure as in the main experiments.

Across all six datasets, increasing k consistently reduces precision, while recall remains high and generally increases as additional candidates are aggregated. Retrieval F1 is highest at $k = 1$ in every setting because the top-ranked candidate provides the most compact support. However, the end-to-end experiments prioritize evidence completeness over retrieval F1. At $k = 3$, recall is at least 0.945 in every evaluated setting, while increasing the budget to $k = 5$ improves recall by at most 0.013 and

reduces precision in every dataset. We therefore use $k = 3$ as a fixed high-recall compromise between evidence completeness and support compactness, rather than as an F1-optimal setting.

B.5. Error Analysis

We distinguish three principal categories: *missed evidence*, where the retrieved support omits at least one required evidence unit; *over-retrieval*, where the complete ground-truth support is retrieved together with additional units; and *downstream answer failure*, where the required support is available but the final answer is incorrect.

Text-based QA. We analyze SAGE-QA Text-Chain errors on HotpotQA and 2WikiMultiHopQA. Complete retrieval failure is limited: 3.3% of HotpotQA examples miss the ground-truth support entirely, while no 2WikiMultiHopQA examples are classified as *retrieval-missed-ground-truth-support* in this analysis. This indicates that the top-3 retrieved candidates usually contain the evidence needed to answer the question.

The dominant pattern is over-retrieval. On HotpotQA, 64.2% of examples fall into the *correct answer, over-retrieval* category; on 2WikiMultiHopQA, the corresponding rate is 54.7%. In these cases, the answer is correct, but the predicted support contains additional irrelevant evidence and therefore fails strict support matching. This is also reflected in support sizes: HotpotQA predictions contain 4.80 evidence units on average compared with 2.46 ground-truth support units, while 2WikiMultiHopQA predictions contain 4.69 units compared with 2.43 ground-truth support units. The average number of extra support units is 2.56 for HotpotQA and 2.28 for 2WikiMultiHopQA.

A second major error type is reader failure despite retrieved evidence. The LLM fails even when the ground-truth support is available in the top-3 candidates in 24.2% of HotpotQA examples and 26.5% of 2WikiMultiHopQA examples. This shows that support retrieval and answer generation remain distinct sources of error. Overall, the text error analysis suggests that the main bottleneck is not evidence recall alone, but converting retrieved candidates into a compact support set and a reliable answer.

A representative HotpotQA error illustrates the support-compression issue. For the benchmark question *"The lamp used in many lighthouses is similar to this type of lamp patented in 1780 by Aimé Argand?"*, SAGE-QA correctly answers *Argand lamp*. The ground-truth support consists of two sentences, *Lewis lamp*$_0$ and *Argand lamp*$_0$, and both are present in the retrieved support. However, the final top-3 context also includes additional lamp-related sentences, such as *Hollow-cathode lamp*$_0$, *Hefner lamp*$_0$, and *Carbon button lamp*$_0$. Thus, the answer is correct and support recall is high, but support precision is reduced. This example shows that the system often retrieves sufficient evidence, but does not yet compress it into a minimal support set.

Ontology-based QA. For the ontology-based datasets, we analyze SAGE-QA Proof errors at Top-3. The results show that complete retrieval failure is not the main bottleneck. Across Family and OWL2Bench, most examples either have exactly correct support or contain the ground-truth support together with additional evidence. The largest category is consistently *correct answer, over-retrieval*. On Family 1-hop, 33.8% of examples have both the correct answer and correct support, while 56.1% have the correct answer but over-retrieval. On Family 2-hop, the same pattern holds, with 31.5% correct-answer/correct-support examples and 57.3% correct-answer/over-retrieval examples. OWL2Bench shows a similar trend: 56.8% of OWL2Bench 1-hop examples and 50.4% of OWL2Bench 2-hop examples fall into the *correct answer, over-retrieval* category. These results indicate that SAGE-QA Proof often

retrieves sufficient symbolic evidence and predicts the correct answer, but the final support context is frequently larger than the ground-truth support.

This pattern is also reflected in the support sizes. For Family 1-hop, the predicted support contains 4.42 evidence units on average, compared with 1.94 ground-truth support units, yielding 2.49 extra support units and only 0.01 missing support units on average. For Family 2-hop, the predicted support increases to 15.07 evidence units on average, compared with 1.97 ground-truth support units, yielding 13.12 extra support units and 0.02 missing support units on average. OWL2Bench follows the same tendency: predicted supports contain 5.61 units on average for OWL2Bench 1-hop and 12.36 units for OWL2Bench 2-hop, compared with ground-truth support sizes of 1.55 and 1.69, respectively. In both OWL2Bench settings, the average missing support is 0.00. Thus, the principal limitation is over-retrieval rather than missing symbolic evidence, particularly in the 2-hop settings.

A Family 2-hop example illustrates the same support-compression issue in the OWL setting. For the question *"Is Ann Jessop a woman?"*, SAGE-QA Proof correctly predicts *TRUE*. The ground-truth support contains three evidence units: `charles_jessop_1831 hasSister ann_jessop_1833`, `hasSister owl:inverseOf isSisterOf`, and `isSisterOf domain Woman`. These units are sufficient to infer that Ann Jessop is a woman, and all three are retrieved. However, the final support context also includes six additional family-relation facts, including `ann_jessop_1833 hasBrother james_jessop_1836`, `ann_jessop_1833 hasBrother john_jessop_1834`, `ann_jessop_1833 hasSister amelia_jessop_1837`, and `edward_jessop_1802 isFatherOf ann_jessop_1833`. Thus, the answer is correct and the required symbolic proof is present, but the support is complete but non-minimal. This example shows that proof-aware reranking can retrieve sufficient symbolic evidence, while final support compression remains necessary.

B.6. GNN-RAG Adaptation

We evaluate GNN-RAG using the upstream `cmavro/GNN-RAG`[1] implementation, integrated under `third_party/GNN-RAG`. Because the original system is designed for KGQA-style graph inputs, we convert each SAGE-QA dataset into the expected GNN-RAG data format before training and inference.

For each dataset, we construct a GNN-RAG graph in which evidence units are represented as graph nodes and connected to query entities through relation-labeled edges. For OWL datasets, ontology facts and axioms are converted into graph evidence nodes. For HotpotQA and 2WikiMultiHopQA, we adapt the text benchmarks into KGQA-style graph inputs by representing sentence evidence as graph nodes and by adding text-derived KG triples. These triples are serialized as subject–predicate–object graph facts and connected back to sentence evidence nodes when the sentence mentions the corresponding KG entity. For HotpotQA, the KG triples are constructed from the textual context using the LLM-based text-to-KG pipeline. For 2WikiMultiHopQA, we use the dataset-provided evidence triples and augment them with LLM-extracted triples. The constructed KG triples are used only to enrich graph reasoning.

We train the upstream ReaRev retriever with LSTM question encoding on the adapted graph data, using the same number of epochs as in the SAGE-QA retriever experiments. After training, the ReaRev retriever produces `test.info`, which contains the graph candidates selected by GNN-RAG. These candidates are passed to the upstream GNN-RAG LLM prediction module, `predict_answer.py`, using `gpt-4.1-mini` as the LLM. The resulting `predictions.jsonl` file is converted back into the common SAGE-QA evaluation format so that GNN-RAG can be evaluated with the same answer and

[1] https://github.com/cmavro/GNN-RAG

joint QA metrics as the other methods. We do not report candidate-level retrieval results for GNN-RAG, since its retrieved entities and paths include KG bridge entities and relation paths that are not directly comparable to SAGE-QA sentence-support candidate structures.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (OpenAI) for Grammar and spelling check, Paraphrase and reword, Improve writing style, Formatting assistance, and Content enhancement. OpenAI Codex was used for assistance with code debugging and code readability. After using these tools/services, the authors reviewed and edited the generated content and suggestions as needed and take full responsibility for the publication's content.

References

[1] P. Lewis, E. Perez, A. Piktus, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems, 2020.

[2] G. Izacard, E. Grave, Leveraging passage retrieval with generative models for open domain question answering, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, 2021, pp. 874–880.

[3] Z. Yang, P. Qi, S. Zhang, et al., Hotpotqa: A dataset for diverse, explainable multi-hop question answering, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018.

[4] X. Ho, A.-K. D. Nguyen, S. Sugawara, A. Aizawa, Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps, in: Proceedings of the 28th International Conference on Computational Linguistics, 2020.

[5] H. Trivedi, N. Balasubramanian, T. Khot, A. Sabharwal, Musique: Multihop questions via single-hop question composition, Transactions of the Association for Computational Linguistics (2022).

[6] Y. Feldman, R. El-Yaniv, Multi-hop paragraph retrieval for open-domain question answering, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2296–2309. URL: https://aclanthology.org/P19-1222/. doi:10.18653/v1/P19-1222.

[7] R. Das, A. Godbole, D. Kavarthapu, Z. Gong, A. Singhal, M. Yu, X. Guo, T. Gao, H. Zamani, M. Zaheer, A. McCallum, Multi-step entity-centric information retrieval for multi-hop question answering, in: A. Fisch, A. Talmor, R. Jia, M. Seo, E. Choi, D. Chen (Eds.), Proceedings of the 2nd Workshop on Machine Reading for Question Answering, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 113–118. URL: https://aclanthology.org/D19-5816/. doi:10.18653/v1/D19-5816.

[8] V. Yadav, S. Bethard, M. Surdeanu, Unsupervised alignment-based iterative evidence retrieval for multi-hop question answering, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4514–4525. URL: https://aclanthology.org/2020.acl-main.414/. doi:10.18653/v1/2020.acl-main.414.

[9] H. Trivedi, N. Balasubramanian, T. Khot, A. Sabharwal, Interleaving retrieval with chain-of-

thought reasoning for knowledge-intensive multi-step questions, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023.

[10] J. Fang, Z. Meng, C. Macdonald, Trace the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation, in: Findings of the Association for Computational Linguistics: EMNLP 2024, 2024.

[11] H. Liu, Z. Wang, X. Chen, Z. Li, F. Xiong, Q. Yu, W. Zhang, Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation, in: Findings of the Association for Computational Linguistics: ACL 2025, 2025.

[12] M. Ding, C. Zhou, Q. Chen, H. Yang, J. Tang, Cognitive graph for multi-hop reading comprehension at scale, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 2694–2703. URL: https://aclanthology.org/P19-1259/. doi:10.18653/v1/P19-1259.

[13] M. Thayaparan, M. Valentino, V. Schlegel, A. Freitas, Identifying supporting facts for multi-hop question answering with document graph networks, 2019. URL: https://arxiv.org/abs/1910.00290. arXiv:1910.00290.

[14] L. Qiu, Y. Xiao, Y. Qu, H. Zhou, L. Li, W. Zhang, Y. Yu, Dynamically fused graph network for multi-hop reasoning, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 6140–6150. URL: https://aclanthology.org/P19-1617/. doi:10.18653/v1/P19-1617.

[15] Y. Fang, S. Sun, Z. Gan, R. Pillai, S. Wang, J. Liu, Hierarchical graph network for multi-hop question answering, CoRR abs/1911.03631 (2019). URL: http://arxiv.org/abs/1911.03631. arXiv:1911.03631.

[16] N. Shao, Y. Cui, T. Liu, S. Wang, G. Hu, Is Graph Structure Necessary for Multi-hop Question Answering?, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 7187–7192. URL: https://aclanthology.org/2020.emnlp-main.583/. doi:10.18653/v1/2020.emnlp-main.583.

[17] D. Groeneveld, T. Khot, Mausam, A. Sabharwal, A simple yet strong pipeline for HotpotQA, in: B. Webber, T. Cohn, Y. He, Y. Liu (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 8839–8845. URL: https://aclanthology.org/2020.emnlp-main.711/. doi:10.18653/v1/2020.emnlp-main.711.

[18] V. Yadav, S. Bethard, M. Surdeanu, If you want to go far go together: Unsupervised joint candidate evidence retrieval for multi-hop question answering, in: K. Toutanova, A. Rumshisky, L. Zettlemoyer, D. Hakkani-Tur, I. Beltagy, S. Bethard, R. Cotterell, T. Chakraborty, Y. Zhou (Eds.), Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, Online, 2021, pp. 4571–4581. URL: https://aclanthology.org/2021.naacl-main.363/. doi:10.18653/v1/2021.naacl-main.363.

[19] P. Jansen, Multi-hop inference for sentence-level TextGraphs: How challenging is meaningfully combining information for science question answering?, in: G. Glavaš, S. Somasundaran, M. Riedl, E. Hovy (Eds.), Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12), Association for Computational Linguistics, New Orleans,

Louisiana, USA, 2018, pp. 12–17. URL: https://aclanthology.org/W18-1703/. doi:10.18653/v1/W18-1703.

[20] C. Mavromatis, G. Karypis, Gnn-rag: Graph neural retrieval for efficient large language model reasoning on knowledge graphs, in: Findings of the Association for Computational Linguistics: ACL 2025, 2025.

[21] W. Wu, Y. Jing, Y. Wang, W. Hu, D. Tao, Graph-augmented reasoning: Evolving step-by-step knowledge graph retrieval for llm reasoning, 2025. URL: https://arxiv.org/abs/2503.01642. arXiv:2503.01642.

[22] H. Bian, Y. Qi, R. Yang, Y. Che, J. Wang, H. Xia, R. Zhen, From query to logic: Ontology-driven multi-hop reasoning in llms, 2025. URL: https://arxiv.org/abs/2508.01424. arXiv:2508.01424.

[23] R. Stevens, M. Stevens, A family history knowledge base using owl 2, in: OWL: Experiences and Directions, 2008. URL: https://api.semanticscholar.org/CorpusID:10478581.

[24] G. Singh, S. Bhatia, R. Mutharaju, OWL2Bench: A Benchmark for OWL 2 Reasoners, in: J. Z. Pan, V. A. M. Tamma, C. d'Amato, K. Janowicz, B. Fu, A. Polleres, O. Seneviratne, L. Kagal (Eds.), The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part II, volume 12507 of *Lecture Notes in Computer Science*, Springer, 2020, pp. 81–96. URL: https://doi.org/10.1007/978-3-030-62466-8_6. doi:10.1007/978-3-030-62466-8_6.